Detecting Irrelevant subtrees to improve probabilistic learning from tree-structured data

Amaury Habrard, Marc Bernard, Marc Sebban

Fundamenta Informaticae (2004) 1–28
IOS Press

Detecting Irrelevant Subtrees to Improve Probabilistic Learning from Tree-structured Data

Amaury Habrard
EURISE – Université Jean Monnet de Saint-Etienne
23, rue du Dr Paul Michelon
42023 Saint-Etienne cedex 2 – France
amaury.habrard@univ-st-etienne.fr

Marc Bernard
EURISE – Université Jean Monnet de Saint-Etienne
23, rue du Dr Paul Michelon
42023 Saint-Etienne cedex 2 – France
marc.bernard@univ-st-etienne.fr

Marc Sebban
EURISE – Université Jean Monnet de Saint-Etienne
23, rue du Dr Paul Michelon
42023 Saint-Etienne cedex 2 – France
marc.sebban@univ-st-etienne.fr

Abstract. In front of the large increase of the available amount of structured data (such as XML documents), many algorithms have emerged for dealing with tree-structured data. In this article, we present a probabilistic approach which aims at *a priori* pruning noisy or irrelevant subtrees in a set of trees. The originality of this approach, in comparison with classic data reduction techniques, comes from the fact that only a part of a tree (*i.e.* a subtree) can be deleted, rather than the whole tree itself. Our method is based on the use of confidence intervals, on a partition of subtrees, computed according to a given probability distribution. We propose an original approach to assess these intervals on tree-structured data and we experimentally show its interest in the presence of noise.

Keywords: data reduction, tree-structured data, noisy data, stochastic tree automata.

1. Introduction

The use of structured or semi-structured data is increasing in research domains such as knowledge discovery in databases or machine learning. One of the main reasons of this trend is the fact that modern

Address for correspondence: EURISE, Faculté des Sciences, 23 rue du Dr Paul Michelon, 42023 Saint-Etienne cedex 2, France

data are often stored in relational databases that provide a higher expressiveness. Thus, it is in the interest of machine learning and data mining approaches to take advantage of the structure of data in order to infer better classification models. The increasing interest for directly dealing with such data has led to the new so-called multi-relational learning field [13].

While a linear representation is sufficient for treating single relational databases, more complex structures, such as trees or graphs, are required for modeling multi-relational databases. In this paper, we focus on tree-structured data which provide an interesting compromise between graphs and linear representations. Actually, trees permit the expression of hierarchical dependences and are less costly, from a computational point of view, than graphs. A lot of applications can be modelized using tree-based representations such as in medicine and biology. Moreover, two main fields have recently stimulated the interest for tree-structured data. The first one is natural language processing, which consists in building language models from a set of sentences. In this context, the tree representation, called *Treebanks*, allows one to embed the grammatical structure of a sentence. The other field, probably the most active one, concerns applications issued from the World Wide Web. Indeed, tree-structured data are natural candidates to represent the information available on the web, such as XML data. Because of the huge quantity of available information, it becomes necessary to have efficient Machine Learning or Data Mining approaches to deal with such data.

From a data mining standpoint, new approaches have been proposed for specifically extracting knowledge from a set of trees. The probably most interesting trend consists in adapting the well known algorithm *Apriori* [2] to tree-structured data. *Apriori*, proposed by Agrawal and Srikant in 1994, aims at extracting frequent itemsets, that are objects appearing with a significant number of occurrences in a database made of flat representations. In [38], Zaki proposed an adaptation of this approach to extract all frequent subtrees from a forest. In [36], Termier *et al.* proposed a similar work based on an approximation procedure, which decreases the computational cost. Miyaharah *et al.* [29] proposed to extract edge labeled tree patterns. More recently, Nijssen and Kok [31] extended the previous approaches to unordered tree patterns.

From a machine learning point of view, several approaches have been proposed to learn tree patterns. In [18], Goldman *et al.* presented a polynomial algorithm able to learn union of tree patterns from a constant number of patterns. Amoth *et al.* studied an exact inference method to learn unordered tree patterns from a set of unordered trees [3]. Grammatical Inference presents also an interesting framework for learning models from a set of trees. The objective consists in inferring a grammar representing a tree language that corresponds to the learning sample. The grammar can be represented by a tree automaton [11, 16] allowing to define a concept. In this framework, Knuutila [25] proposed to learn finite tree automata from positive and negative learning trees. García and Oncina [15] proposed to learn tree automata from skeleton trees, that are trees without label on internal nodes. To avoid the use of negative examples, Rico *et al.* [33] dealt with the learning of specific tree automata, called k-testable tree automata. Another approach to avoid the use of negative examples consists in learning statistical models. In this context, the objective is to infer stochastic tree automata that allow to define a probabilistic distribution over a set of trees. Carrasco *et al.* [8] proposed an adaptation of previous works of Knuutila [25] and García and Oncina [15] in order to learn stochastic tree automata. Rico *et al.* [34] also extended their own work to learn k-testable stochastic tree automata. Various applications on tree-structured data used the grammatical inference framework based on trees. Abe *et al.* [1] applied stochastic tree automata to predict the secondary structure of proteins represented by trees. Kosala *et al.* [27] used the k-testable approach

to extract knowledge from XML data. Habrard *et al.* [19] proposed a generalization of stochastic tree automata to extract tree patterns and then allow knowledge extraction from medical data [20].

Machine learning and data mining algorithms dealing with structured data have to overcome the same couple of drawbacks than the one imposed by unstructured data. First, in front of modern databases, they require to process huge amounts of data. This algorithmic constraint becomes particularly important in the context of tree-structured data which are obviously harder to process than linear representations. Second, they also have to deal with a high level of noise, which can have dramatic impacts on the quality of the inferred models. In such a context, a classic strategy consists in removing some instances from the dataset which are either irrelevant, because of the weak information they provide, or detected as being noisy. These tasks are the matter of data reduction which can be achieved via two ways: prototype selection (PS) [37] and feature selection (FS) [22]. It is important to note here that these methods usually require counter-examples and aim at totally deleting either an example (PS) or a feature (FS). Originally proposed to tackle the problem of storage requirements of case-based learning algorithms, such as k-Nearest-Neighbors [12], data reduction techniques have rarely been applied, so far, to structured data.

In this paper, we focus more specifically on the ability of data reduction techniques to deal with irrelevant or noisy instances. We propose an approach allowing to detect such data in a set of trees. In this context, we can assume that only some particular subtrees of a given tree are noisy or irrelevant and then deserve to be removed. The suppression of a subtree can be seen as a hybrid approach of data reduction. It looks like prototype selection when a tree is completely deleted, and it looks like a local feature selection when only subtrees are removed. Recent complexity theoretic results show that the problems of PS and FS are NP-hard [32]. This advocates of the use of heuristics for selecting relevant instances in a set of trees.

The goal of our approach is to improve the learning of probabilistic models from a set of trees in the presence of noise. In the probabilistic framework, the objective is not to learn a classifier which is able to discriminate positive from negative examples, but to learn a probability distribution over the data. A good probabilistic model gives a correct estimation of the probability of each example. Such models are usually inferred from a set of positive examples only, which can be useful for many real world applications. However the presence of irrelevant or noisy data can dramatically affect the quality of the inferred distribution. The goal of this paper is to provide an approach allowing us to deal with such data in the framework of probabilistic learning. In fact, our main objective is not to highly reduce the size of the learning set, but rather to improve the probability estimations of stochastic models in order to increase their predictive ability.

To achieve this goal, we propose a probabilistic data reduction approach specifically adapted to tree-structured data. The method is based on a partitioning of the whole set of subtrees, using on regular tree patterns (or contexts), and on the evaluation of the relevance of the probability of a subtree to be in a given partition. Subtrees with a too small probability are deleted. This task is carried out thanks to a confidence interval computed from the subtrees belonging to the partition. We decided to evaluate our approach in the context of learning stochastic tree automata. These models allow us to define a probability distribution on a set of trees using the representation of a probabilistic automata. Such automata define a probabilistic tree language and are learned using statistical information from a learning set constituted of trees. These automata are then very suited for learning a probability distribution over a set of trees.

The paper is organized as follows. Section 2 deals with some definitions and notations. In Section 3, we introduce our data reduction approach. Section 4 presents a theoretical analysis of the impact of noise. In Section 5, we describe our experimental results. Finally, we conclude this paper in Section 6.

2. Definitions and Notations

In this section, we formally define the notion of tree, and we illustrate our definitions using an example presented in Figure 1. In this example, that we will use as illustration all along this paper, we consider a database of hospital patients affected by a given disease. For each patient, we have two of the main characteristic symptoms of the disease, his blood group and the stage of the disease.

The definitions introduced in this section correspond to a classic framework, and are inspired by the thesis of Kilpeläinen [23].

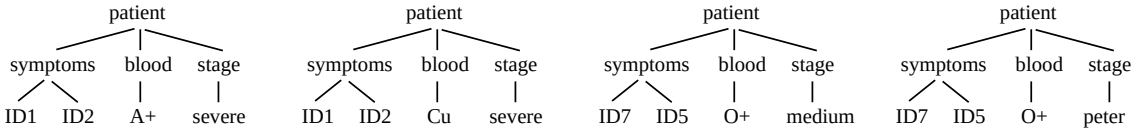


Figure 1. Set of trees T representing 4 patients.

Definition 1. A binary relation R on a set D is a subset of the Cartesian product $D \times D$

Definition 2. The transitive closure of R, denoted by R^+, is defined by:

$$\begin{aligned} R^1 &= \{(x,y) \mid (x,y) \in D^2\} \\ R^{n+1} &= \{(x,y) \mid \exists z \in D \text{ such that } (x,z) \in R \text{ and } (z,y) \in R^n, n > 0\} \\ R^+ &= \bigcup_{n>0} R^n \end{aligned}$$

Definition 3. A tree t is defined by a triplet $(N_t, A_t, root(t))$ where N_t is a set of nodes, $A_t \subseteq N_t^2$ is a binary relation, and $root(t)$ is a special node called the **root** of the tree t. For each pair $(u,v) \in A_t$, (u,v) is called an edge of the tree and u is the parent of v, which is denoted by $u = parent(v)$. A_t must verify the following conditions:

- $root(t)$ has no parent.
- Each node in t (except the root) has exactly one parent.
- From the root, we can reach any node of t in following a path defined by the edges of t, *i.e.* $\forall v \in N_t, v \neq root(t), (root(t), v) \in A_t^+$.

Let t be a tree and let $u \in N_t$, we now formally define the notion of children and descendants of a node u.

Definition 4. The children of u are defined by

$$children(u) = \{v \in N_t \mid (u,v) \in A_t\}$$

Definition 5. The descendants of u by:

$$descendants(u) = \{v \in N_t \mid (u, v) \in A_t^+\}$$

Definition 6. The subtree of t with root u is the tree $(N_{t[u]}, A_{t[u]}, u)$ such that:

- $N_{t[u]} = \{u\} \cup descendants(u)$
- $A_{t[u]} = A_t \cap (N_{t[u]} \times N_{t[u]})$

Trees, we are interested in, are constructed over a signature. This signature allows to label each node by a functional symbol, such that all the nodes labeled by the same symbol have exactly the same number of children. Finally, the symbols are typed and the children ordered.

Definition 7. A *signature* Σ is a 4-tuple $(\tau, V, arity, \sigma)$ where:

- τ is a finite set whose elements are called *sorts*,
- V is a finite set whose elements are called function symbols, V called the alphabet,
- $arity$ is a mapping function from V into $\mathbb{N}$, $arity(f)$ called the *arity* of the function symbol f,
- σ is a mapping function from V into τ, $\sigma(f)$ called the *sort* of f.

We denote by Σ^T the set of trees defined relatively to a *signature* Σ.

Definition 8. A labeled tree t over a signature Σ is a tree such that each node $u \in N_t$ is mapped with a symbol $f \in V$ such that $|children(u)| = arity(f)$

We use the notation $t = f(t_1, \ldots, t_n)$ to define a tree having the root labeled by the symbol f and having n subtrees $t_1, \ldots, t_n$ such that $children(root(t)) = \{root(t_1), \ldots, root(t_n)\}$. A total order is defined on the set of children of t such that t_1 is the first subtree and t_n is the last one.

For example, the first of the four trees of Figure 1 represents a patient having symptoms $ID1$ and $ID2$, with $A+$ as blood group and a disease at a severe stage. The label of the root is $patient$, and the labels of the three children of the root are $symptoms$, $blood$ and $stage$. We denote this tree by $patient(symptoms(ID1, ID2), blood(A+), stage(severe))$.

In the following, when there is no ambiguity, we will describe the node of a tree by its label.

Definition 9. A *position* is a couple (f, p) (denoted by $f.p$) where $f \in V$ and $p \in \mathbb{N}$ such that $1 \leq p \leq arity(f)$. A *position* defines the subtree corresponding to the child number p of the symbol f (the children are ordered from left to right). The special position p_{root} defines the symbol at the root of a tree.

For example, in the first tree of Figure 1, the subtree $symptoms(ID1, ID2)$ is at position $patient.1$ of the tree $patient(symptoms(ID1, ID2), blood(A+), stage(severe))$ and the subtree $ID2$ is at position $symptoms.2$ of the subtree $symptoms(ID1, ID2)$.

Definition 10. Let T be a sample of trees. We denote by $Sub(T)$ the set of subtrees of T and $MSub(T)$ the multi-set of subtrees of T.

For example, if we consider the subset composed of the first two trees of the sample T of Figure 1:

$$\begin{aligned} Sub(T) &= \{ID1, ID2, A+, Cu, severe, symptoms(ID1, ID2), blood(A+), blood(Cu), \\ &\quad stage(severe), patients(symptoms(ID1, ID2), blood(A+), stage(severe)), \\ &\quad patients(symptoms(ID1, ID2), blood(Cu), stage(severe))\} \\ MSub(T) &= \{ID1, ID1, ID2, ID2, A+, Cu, severe, severe, symptoms(ID1, ID2), \\ &\quad symptoms(ID1, ID2), blood(A+), blood(Cu), stage(severe), stage(severe), \\ &\quad patients(symptoms(ID1, ID2), blood(A+), stage(severe)), \\ &\quad patients(symptoms(ID1, ID2), blood(Cu), stage(severe))\} \end{aligned}$$

Let us now consider the problem of noisy data. In our example, the subtree $blood(Cu)$ of the second tree and the subtree $stage(peter)$ of the fourth one, are clearly corrupted, probably due to a typing error, and should be removed. However, the other subtrees of these trees deserve to be kept because they bring *relevant* information. One of the interest of a data reduction approach is to reduce the dataset size by removing useless or noisy data without calling into question the probabilistic distribution of the learning set. Our idea is then to assess this distribution and remove data having a too weak probability density, that characterizes both useless and noisy data. This leads us to propose a definition of the relevance of a subtree. This definition is in relation with the one of relevant features introduced in [22].

Definition 11. Let D be a distribution on a set of subtrees S, a subtree t is relevant in S if and only if its probability estimation $\hat{p}_D(t)$ is not *significantly* smaller than the mean of the probabilities of subtrees of S.

Roughly speaking, this definition expresses the idea that a relevant subtree must cover a significant part of the probability density of the learning set. Then, to assess this notion of significance, we propose, in the next section, to use the concept of confidence intervals, widely used in statistical inference theory.

3. Pruning Subtrees using Confidence Intervals

In this section, we introduce the probabilistic framework of our approach. First, we compute a probability distribution over a set of trees, based on "N-grams" models [30] and adapted to tree-structured data. Using this distribution, we draw a confidence interval around the mean of the probabilities of subtrees. The lower bound of this interval is then used as a critical threshold for testing the relevance of a subtree.

All this process requires to take into account, in the considered learning set, only comparable subtrees. Actually, while $blood(A+)$ and $blood(O+)$ are two subtrees characterizing the same "concept", it is obviously irrelevant to compare $blood(A+)$ and $stage(severe)$. In other words, it means that one must formally define a notion of *concept* for permitting the construction of efficient confidence intervals. We present this formalism at the end of this section, based on a partitioning method using regular tree patterns.

3.1. Tree-based Probability Distribution

Given a learning sample T of trees, we aim at constructing a probability distribution on the whole set T. To achieve this task, we compute a probability for each subtree in $Sub(T)$ according to an approach

similar to the one of "N-grams" often used in natural language modeling [6]. This approach assumes that the probability of a given symbol in a string can be computed using the $n-1$ previous symbols. We use a similar principle, with $n=2$, computing the probability of a symbol relatively to its parent. Note that a different adaptation of this approach in the context of trees has been proposed in [34]. For each symbol a of the learning set T, we assess its probability $\hat{p}_c(a \mid f.i)$ to be the child number i of any symbol f. Formally, if $\#^T_{f.i}(a)$ is the number of nodes labeled by a at position $f.i$ in the set T and if $\#^T(f)$ corresponds to the number of nodes labeled by f in T, we have:

$$\forall a \in V, \forall f \in V, \forall 1 \leq i \leq arity(f), \hat{p}_c(a \mid f.i) = \frac{\#^T_{f.i}(a)}{\#^T(f)}$$

Moreover we compute for each symbol its probability to be the root of a tree.

$$\forall a \in V, \hat{p}_r(a) = \frac{\#^T_{p_{root}}(a)}{|T|}$$

where $|T|$ is the cardinality of the set T.
Finally the probability of a tree $t = f(t_1, \ldots, t_n)$ is computed as follows:

$$\hat{p}_a(f(t_1, \ldots, t_n)) = \hat{p}_r(f) \times \hat{p}_{pos}(t_1 \mid f.1) \times \cdots \times \hat{p}_{pos}(t_n \mid f.n)$$

where $\hat{p}_{pos}$ is recursively defined by:

$$\hat{p}_{pos}(g(u_1, \ldots, u_m) \mid f.i) = \hat{p}_c(g \mid f.i) \times \hat{p}_{pos}(u_1 \mid g.1) \times \cdots \times \hat{p}_{pos}(u_m \mid g.m)$$

For example, the set of trees of Figure 1 leads to the following conditional probabilities (only the non null probabilities are indicated):

$\hat{p}_c(ID1 \mid symptoms.1) = \frac{2}{4}$ $\quad$ $\hat{p}_c(A+ \mid blood.1) = \frac{1}{4}$ $\quad$ $\hat{p}_c(medium \mid stage.1) = \frac{1}{4}$
$\hat{p}_c(ID2 \mid symptoms.2) = \frac{2}{4}$ $\quad$ $\hat{p}_c(Cu \mid blood.1) = \frac{1}{4}$ $\quad$ $\hat{p}_c(peter \mid stage.1) = \frac{1}{4}$
$\hat{p}_c(ID7 \mid symptoms.1) = \frac{2}{4}$ $\quad$ $\hat{p}_c(O+ \mid blood.1) = \frac{2}{4}$ $\quad$ $\hat{p}_c(stage \mid patient.3) = \frac{4}{4}$
$\hat{p}_c(ID5 \mid symptoms.2) = \frac{2}{4}$ $\quad$ $\hat{p}_c(severe \mid stage.1) = \frac{2}{4}$ $\quad$ $\hat{p}_c(blood \mid patient.2) = \frac{4}{4}$
$\hat{p}_c(symptoms \mid patient.1) = \frac{4}{4}$ $\quad$ $\hat{p}_r(patient) = \frac{4}{4}$

and the probability of the tree $t = patient(symptoms(ID1, ID2), blood(A+), stage(severe))$ is computed as follows:

$$\begin{aligned}
\hat{p}_a(t) &= \hat{p}_r(patient) \times \hat{p}_c(symptoms \mid patient.1) \times \hat{p}_c(blood \mid patient.2) \times \\
&\quad \hat{p}_c(stage \mid patient.3) \times \hat{p}_c(ID1 \mid symptoms.1) \times \hat{p}_c(ID2 \mid symptoms.2) \times \\
&\quad \hat{p}_c(A+ \mid blood.1) \times \hat{p}_c(severe \mid stage.1) \\
&= 1 \times 1 \times 1 \times 1 \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{4} \times \frac{1}{2} = \frac{1}{32}
\end{aligned}$$

If t' is a subtree of a tree, assuming that t' is different than the whole tree, then we compute its probability taking into account its position relatively to its parent. For example, the probability of the subtree

$blood(A+)$ of the previous tree t is computed as follows:

$$\hat{p}_{pos}(blood(A+) \mid patient.2) = \hat{p}_c(blood \mid patient.2) \times \hat{p}_c(A+ \mid blood.1) = \frac{4}{4} \times \frac{1}{4} = \frac{1}{4}$$

The probability distribution allows us to compute the lower bound of a confidence interval which helps us to detect irrelevant instances. However, when a confidence interval is estimated with a small number of examples, the lower bound can be negative. It is then impossible to detect very rare instances. To overcome this drawback, we propose to slightly modify the probability distribution in order to automatically delete very rare instances, *i.e.* that occur only once. For this purpose, to compute the probability of a subtree t, we adapt the probability distribution such that the instance of t is not taken into account. Formally, if $\hat{p}_c(a|f.i)$ is involved in the computation of the probability estimation of t, then $\hat{p}_c(a|f.i)$ is replaced by:

$$\begin{cases} 0 & \textbf{if } \#^T(f) = \#^t(f) \\ \frac{\#^T_{f.i}(a) - \#^t_{f.i}(a)}{\#^T(f) - \#^t(f)} & \textbf{otherwise} \end{cases}$$

where $\#^t_{f.i}(a)$ is the number of symbols a at position $f.i$ in t, and $\#^t(f)$ the number of labels f in t.

For example, the probability of the subtree $blood(A+)$ is actually evaluated as follows:

$$\hat{p}_{pos}(blood(A+) \mid patient.2) = \hat{p}_c(blood \mid patient.2) \times \hat{p}_c(A+ \mid blood.1) = \frac{3}{3} \times \frac{0}{3} = 0$$

This modification allows us to associate a null probability to instances occurring once in the dataset. In our approach, all subtrees with a null probability are automatically removed.

3.2. Probabilistic Pruning Rule

In the previous section, we have proposed a way to construct a probability distribution from a set of trees T. We show here how we can use this distribution to *a priori* (*i.e.* before any learning process) prune subtrees considered statistically irrelevant, thanks to a statistical test. Let us consider a subset S of $MSub(T)$. We compute a confidence interval, according to a risk α (called the level of significance of the test, or Type I error). We look for an interval $[pmin; 1]$ such that the theoretical probability $p_a(t)$ of a subtree satisfies the following constraint:

$$p(p_a(t) \geq pmin) = 1 - \alpha.$$

According to the Central Limit Theorem, the mean $\overline{\hat{p}_a(t_S)}$ of probabilities of the elements of S follows a normal distribution with an expected value μ and a standard deviation $\frac{\sigma}{\sqrt{|S|}}$. Then the confidence interval around μ is defined as follows:

$$\mu \in \overline{\hat{p}_a(t_S)} \pm \frac{\hat{\sigma}}{\sqrt{|S|}} \times u_\alpha$$

where $\hat{\sigma}$ is the estimated standard deviation and u_α the $(1-\alpha)$-percentile of the normal distribution. Let $pmin$ be the lower bound of this interval, such that:

$$pmin = \overline{\hat{p}_a(t_S)} - \frac{\hat{\sigma}}{\sqrt{|S|}} \times u_\alpha$$

When the size of S is not sufficient ($|S| < 30$) for approximating the distribution of the random variable $p_a(t)$ by the normal distribution, $p_a(t)$ follows a Student distribution with $|S| - 1$ degrees of freedom.

Once this lower bound is computed, our decision rule consists in deleting all subtrees of S that have a probability smaller than $pmin$. Then, we can refine Definition 11 on the relevance of a subtree taking into account this notion of confidence interval.

Definition 12. Let D be a distribution on a set of subtrees S, and let α be a risk. A subtree t is $(1-\alpha)$-relevant in S if and only if $\hat{p}_a(t) \geq \overline{\hat{p_a}(t_S)} - \frac{\hat{\sigma}}{\sqrt{|S|}} \times u_\alpha$.

We claim that such pruning strategy is particularly suited for removing noisy subtrees. In fact, such data have by nature an individual weak probability. Actually, in the opposite case, they would constitute a part of the concept to learn, and then would deserve to be kept in the learning set.

3.3. Partitioning Subtrees with Regular Tree Patterns

So far, we assumed that we have a learning sample T of trees, from which we are now able to detect irrelevant data. However, we think that the previous pruning strategy is relevant if the probability of a subtree is computed relatively to those characterizing the same concept. Roughly speaking, we must compare only what is comparable. In the definitions of Section 2, we considered that children of a node are ordered and we proposed a definition for the position of a subtree. Then, we are able to define subtrees representing the same concept as subtrees that appear exactly at the same position. Indeed, because they appear at the same position, they represent instances of the same concept. In the example of hospital patients (see Figure 1), subtrees at position $patient.3$ are instances of the concept representing the stage of the disease.

In fact, we can go farther in our definition of a concept. With the position of a subtree, we can also take into account some information given by subtrees appearing at other positions. In our example on patients, subtrees at position $patient.2$ define the concept *blood group* of a person. We can refine this context by taking into account subtrees appearing at positions $patient.1$ and $patient.3$. For example, we can consider blood groups of patients having the symptoms $ID1$ and $ID2$ and being at a severe stage of the disease. Subtrees corresponding to this concept of *blood group* are those appearing at position $patient.2$ with the subtree $symptoms(ID1, ID2)$ at position $patient.1$ and the subtree $stage(severe)$ at position $patient.3$. In this case, a concept is defined by a *context* taking into account the maximum of local information available.

To assess this notion of context, we propose a method of partitioning using regular tree patterns, *i.e.* tree patterns with only one variable. This approach ensures that two subtrees that appear in the same context (*i.e.* subtrees with the same ancestors and siblings) will be in the same partition. This kind of tree patterns corresponds exactly to our notion of context. In fact, we can easily construct such tree patterns by replacing only one subtree in a tree by a variable. If we iterate this process in order to consider all

the nodes of T, we can construct all definable contexts in this set. Before presenting our partitioning method, we give a formal definition of regular tree patterns.

Definition 13. A regular tree pattern is a tree t defined on a signature $(\tau, V \cup \{X\}, arity, \sigma)$ where X is a variable and t has exactly one leaf labeled by X.
Let t be a regular tree pattern and t' be a classic tree. We denote by $t._{\#}t'$ the substitution of the variable X of t by the tree t'.

Definition 14. The set of all the regular tree patterns, that are tree patterns with exactly one variable, definable on a signature $\Sigma = (\tau, V \cup \{X\}, arity, \sigma)$ by Σ_T^X.

For example, Figure 2 shows the result of the concatenation of the subtree $symptoms(ID1, ID2)$ with the tree pattern $patient(X, blood(A+), stage(severe))$.

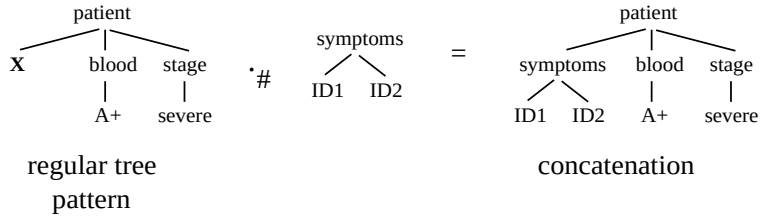


Figure 2. Regular tree pattern and concatenation.

To construct a partition of the multi-set of subtrees, our approach consists in extracting all the regular tree patterns definable from T. As seen before, the number of such tree patterns corresponds exactly to the number of nodes of T. Then formally, the set of all regular tree patterns can be defined by $\{t \mid \exists t' \in MSub(T) \text{ and } t._{\#}t' \in T\}$. Figure 3 shows all the tree patterns definable from the tree $patient(symptoms(ID1, ID2), blood(A+), stage(severe))$.

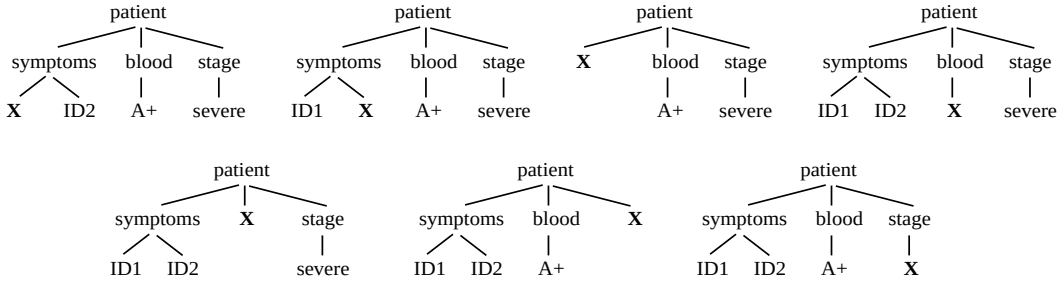


Figure 3. Tree patterns definable from $patient(symptoms(ID1, ID2), blood(A+), stage(severe))$.

Each pattern t allows us to define a class π_t of the partition of the multi-set of subtrees. The subtrees which can be concatenated to t to obtain a tree of the learning sample belong to this partition: $\pi_t = \{t' \in MSub(T) \mid t._{\#}t' \in T\}$. We construct all partitions definable from a set of trees with a quadratic time complexity in the number of nodes in T. This time complexity gives an upper bound of the complexity of our approach. Figure 4 shows examples of the sample t selected by the tree pattern

$patient(symptoms(ID1, ID2), X, stage(severe))$. Note that the subtree $blood(O+)$ of the third tree is not selected because the subtrees at position $patient.3$ in this tree and in the regular tree pattern are different.

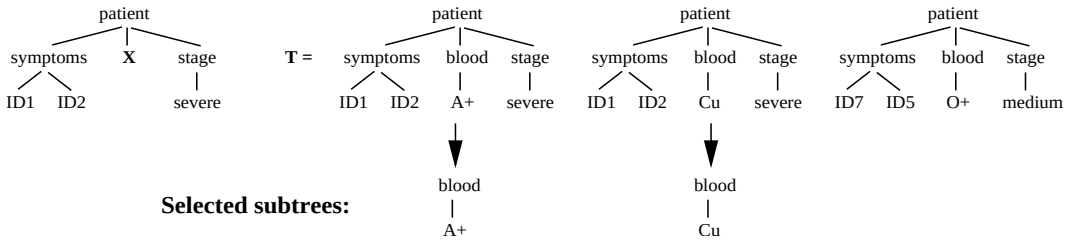


Figure 4. Construction of a partition.

To end this section, we synthesize the main steps of our pruning method in Algorithm 1.

Data: T: a set of trees
α: a real $\in [0; 1]$
begin
  Construct the probability distribution over T
  $T \leftarrow$ partitioning $MSub(T)$
  foreach $S \in T$ **do**
    Compute an interval $[pmin; 1]$ for probabilities of subtrees of S to the risk α
    foreach $t \in S$ **do**
      Delete t if it is not $(1 - \alpha)$-relevant or if $\hat{p}_a(t) = 0$
    end
  end
end

Algorithm 1: Pruning subtrees using confidence intervals.

4. Theoretical Results in the Presence of Noise

As we said before, an important aspect of our work is its capacity to deal with noisy data. In this section, we propose to draw a theoretical study of the impact of noise on our method. To begin with, we define the protocol of the data corruption process we consider. Then, under reasonable assumptions about the nature of noise, we study the impact of noisy data on the probability distribution we have to build. We show that it has a direct influence on the number of deleted subtrees. We finish this study with the examination of the impact of noise on subtrees. From a global point of view, this study has the objective to point out the kind of data that can be efficiently preprocessed in the presence of noise.

We assume that the noise is uniformly distributed on the leaves of the learning set T. We think that such assumption is reasonable in the context of tree-structured data because, in most of applications, the precise information concerning the concept is often stored in leaves (for example $A+$, $severe$), while the other nodes characterize more global information. Let γ be the level of noise, which corresponds to

the percentage of leaves that have been corrupted. We assume that a noisy leaf is replaced by a different leaf, randomly chosen from the other leaves of the dataset. In the following, we denote by L_T the whole set of leaves in T (without repetitions).

According to our experimental set up, only probabilities of symbols located at the leaves, that we will call "constant", are affected by the noise. Then, we will only consider probabilities $p(a|f.n)$ where a is such a constant and f a symbol such that $1 \leq n \leq arity(f)$. In all the section, we assume that for any position $f.n$, $\sum_{a \in V,\ arity(a)=0} p(a|f.n) = 1$.

In the presence of an uniformly distributed noise γ, we can easily determine the probability of any constant a given any position $f.n$:

$$\begin{aligned} P_\gamma(a|f.n) &= (1-\gamma) * p(a|f.n) + (1 - p(a|f.n)) * \tfrac{\gamma}{|L_T|-1} \\ &= (1 - |L_T| * p(a|f.n)) * \tfrac{\gamma}{|L_T|-1} + p(a|f.n) \end{aligned}$$

$P_\gamma(a \mid f.n)$ is composed of two terms. The first one represents the weighted proportion of leaves a that are not affected by the noise, while the second corresponds to the weighted proportion of corrupted data that were originally different from a before their modification into a (due to the noise). An important point to remark is that $P_\gamma(a \mid f.n)$ evolutes linearly in function of γ.

Let us now study the behavior of $P_\gamma(a|f.n)$.

Theorem 4.1. $P_\gamma(a|f.n)$ is an increasing function of γ if $p(a|f.n) < \frac{1}{|L_T|}$ and a decreasing function of γ if $p(a|f.n) > \frac{1}{|L_T|}$

Proof:

$$P_\gamma(a|f.n) = (1 - |L_T| * p(a|f.n)) * \frac{\gamma}{|L_T| - 1} + p(a|f.n)$$

then

$$\frac{\partial P_\gamma(a|f.n)}{\partial \gamma} = (1 - |L_T| * p(a|f.n)) * \frac{1}{|L_T| - 1}$$

Then it is easy to see that $P_\gamma(a|f.n)$ increases with the level of noise when $p(a|f.n) < \frac{1}{|L_T|}$, and decreases when $p(a|f.n) > \frac{1}{|L_T|}$. □

This result shows that when the noise level increases, the probability of the constant a (whatever its nature) tends to the same value $\frac{1}{|L_T|}$.

The number of leaves plays an important role in the estimation of the distribution. We have experimentally observed that, for a given level of noise, the increase of the number of leaves tends to highly disturb the original distribution, that may have a direct impact on our pruning method. To confirm this phenomenon, we theoretically study, here, the deviation between $p(a|f.n)$ and $P_\gamma(a|f.n)$.

Theorem 4.2. The deviation $D_{L_T}(a \mid f.n) = p(a|f.n) - P_\gamma(a|f.n)$ is an increasing function in $|L_T|$.

Proof:

$$\begin{aligned} D_{L_T}(a \mid f.n) &= p(a|f.n) - P_\gamma(a|f.n) \\ &= p(a|f.n) - (1-\gamma) * p(a|f.n) - (1 - p(a|f.n)) * \frac{\gamma}{|L_T|-1} \\ &= \gamma * p(a|f.n) - (1 - p(a|f.n)) * \frac{\gamma}{|L_T|-1} \end{aligned}$$

Then

$$\frac{\partial D_{L_T}(a \mid f.n)}{\partial L_T} = -\frac{-(1-p(a|f.n)) * \gamma}{(|L_T|-1)^2} = \frac{(1-p(a|f.n)) * \gamma}{(|L_T|-1)^2} \geq 0$$

which is positive. □

The deviation between the two probabilities increases with $|L_T|$, and implies that the probability distribution is highly modified, whatever the level of noise γ. This result can be easily interpreted. Indeed, when the number of leaves is high, those originally labeled by a are replaced by others, and at the same time, there is only a small proportion of leaves different from a modified into a. Then, given a level of noise, the probability $p(a|f.n)$ is modified with the increase of $|L_T|$.

Since the number of occurrences of a given tree is used for drawing the probability distribution, we aim here at computing the probability to modify this number according to the level of noise γ. Let N_t be the set of trees having the same number of leaves as t and differing from it in at least one leaf. In other words, for each tree $t' \in N_t$, we can define a bijection between the leaves of t' and the ones of t such that we can obtain t from t'. Given a tree $t' \in N_t$, let $diff(t, t')$ be the number of different leaves between t and t'. We denote by NL_t the number of leaves of a tree t, and by NO_t the number of occurrences of a tree t in the set T.

Theorem 4.3. The probability that the number of occurrences of a tree t is modified by γ is

$$(1-(1-\gamma))^{NL_t} * \frac{NO_t}{|T|} + \sum_{t' \in N_t} \left(\left(\gamma * \frac{1}{|L_T|-1} \right)^{diff(t',t)} * (1-\gamma)^{(NL_t - diff(t',t))} * \frac{NO_{t'}}{|T|} \right)$$

Proof:
Noise may corrupt the number of occurrences of a tree t via two ways. On the one hand, it can affect the leaves of t, on the other hand it can corrupt a tree of N_t which becomes an instance of t.

The probability to corrupt at least one leaf of t is equal to the complementary of the probability not to add noise to any leaf of t, that is $1-(1-\gamma)^{NL_t}$. This quantity requires then to be multiplied by the probability to have the tree t in the set T: $\frac{NO_t}{|T|}$.

Moreover, for any tree $t' \in N_t$, if $diff(t, t') = n$, then we have to add noise to n leaves of t' such that each of them becomes exactly equal to the one of t, and we do not corrupt the other leaves. The probability that t' becomes equal to t is $(\gamma * \frac{1}{|L_T|-1})^{diff(t,t')} * (1-\gamma)^{(NL_t - diff(t,t'))}$. We must multiply this probability by the one of having t': $\frac{NO_{t'}}{|T|}$. We repeat this for each tree in N_t and holds

$$\textstyle\sum_{t' \in N_t} (\gamma * \frac{1}{|L_T|-1})^{diff(t,t')} * (1-\gamma)^{(NL_t - diff(t,t'))} * \frac{NO_{t'}}{|T|}.$$

Then, at the end of the corruption, the probability that the number of occurrences of a tree t is modified by γ is

$$(1-(1-\gamma))^{NL_t} * \frac{NO_t}{|T|} + \sum_{t' \in N_t} \left(\left(\gamma * \frac{1}{|L_T|-1} \right)^{diff(t,t')} * (1-\gamma)^{(NL_t - diff(t,t'))} * \frac{NO_{t'}}{|T|} \right)$$

□

5. Evaluation in the Context of Learning Stochastic Tree Automata

We present, in this section, experimental results which justify the interest of our method to preprocess the learning sample in presence of noisy data. Since we work on trees, we propose to evaluate our data reduction approach in the framework of learning stochastic tree automata from a learning set [8, 15, 35]. In such a context, we have a sample of trees, supposed to be generated from a probability distribution. The objective is to learn the probabilistic model which has generated the data. Thus, the main goal of such an approach is not to learn a classifier which can discriminate negative from positive examples, but rather to learn a statistical distribution over the learning sample. We propose to compare the automata inferred from noisy data and those induced after our pruning process. To carry out this task, we can achieve two series of experiments. The first one deals with learning problems where the target automaton is *a priori* known. In this case, we can use a measure of distance between the inferred model and the target automaton. However, since the target model is often unknown, in a second series of experiments, we also evaluate our approach using a perplexity measure. This criterion assesses the relevance of the model on a test sample.

In this section, we first specify automata we work on, called many-sorted stochastic tree automata, defined relatively to a signature. Then, we present the algorithm we use for inferring stochastic tree automata (STA). Finally, after having detailed the main criteria allowing us to assess the efficiency of our pruning method, we carry out a large experimental study showing the interest of our approach.

5.1. Stochastic Many-Sorted Tree Automata

Deterministic Tree Automata (DTA) generalize Deterministic Finite-state Automata (DFA). In contrast to DFA, that parse strings from left to right, DTA work bottom-up. A state of the automaton is associated to each node of the tree. The label of each node is defined by a transition function, and the state of the root determines if the tree belongs to the language or not. To illustrate this informal description, we give a graphical representation of such automata in Figure 5. In this example, states are represented by a circle, final states by a double circle and transitions by a triangle containing the symbol involved in the transition. Let us define formally the concept of stochastic tree automata able to recognize trees defined on a signature Σ (already presented in Section 2).

Definition 15. A SMDTA is a 5-tuple $A = (\Sigma, Q, r, \delta, p)$ where

- Σ is a signature $(\tau, V, arity, \sigma)$,
- $Q = \cup_{s \in \tau} Q^s$ is a finite set of states, each state having a sort in S,
- $r : Q \longrightarrow [0, 1]$ is the probability for a state to be a final state,
- $\delta : V \times Q^* \longrightarrow Q$ is the transition function,
- $p : V \times Q^* \longrightarrow [0, 1]$ is the probability of a transition.

The transition function δ is recursively extended to a function δ': $V \times (\Sigma^T)^* \longrightarrow Q$ as follows:

$$\begin{cases} \delta'(f) = \delta(f) \textbf{ if } arity(f) = 0 \\ \delta'(f(t_1, \ldots, t_n)) = \delta(f, \delta'(t_1), \ldots, \delta'(t_n)) \textbf{ otherwise} \end{cases}$$

Note that we consider deterministic automata with the following properties. The transition rules of δ are valid, that is for each rule $(f, q_1, \ldots, q_n) \longrightarrow q$, $arity(f) = n$ and $\sigma(f) = \sigma(q)$. Probabilities are normalized, so that probabilities of transitions leading to the same state sum to one.

The probability of a tree t parsed by a tree automaton A is computed as follows:

$$p(t \mid A) = r(\delta'(t)) \times \pi(t)$$

where $\pi(t)$ is recursively given by:

$$\pi(f(t_1, \ldots, t_n)) \quad = \quad p(f, \delta'(t_1), \ldots, \delta'(t_n)) \times \pi(t_1) \times \cdots \times \pi(t_n)$$

$\pi(f)$ stops when f is a constant symbol, that is when $arity(f) = 0$. We say that an SMDTA A accepts a tree t if and only if $p(t \mid A) > 0$. The language recognized by an automaton is the set of all trees accepted by the automaton.

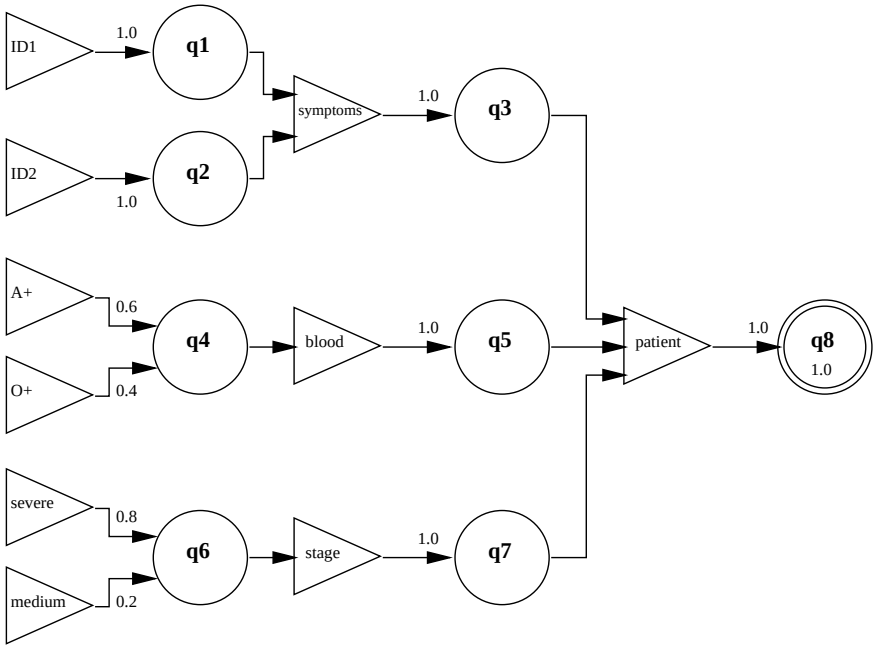


Figure 5. An example of stochastic tree automaton.

For example, let us consider the automaton A in Figure 5 which is defined with the following transitions and probabilities:

$$\begin{array}{ll}
1.0 : ID1 \longrightarrow q_1 & 0.6 : A+ \longrightarrow q_4 \\
1.0 : ID2 \longrightarrow q_2 & 0.4 : O+ \longrightarrow q_4 \\
1.0 : symptoms(q_1, q_2) \longrightarrow q_3 & 1.0 : blood(q_4) \longrightarrow q_5 \\
0.2 : medium \longrightarrow q_6 & 0.8 : severe \longrightarrow q_6 \\
1.0 : stage(q_6) \longrightarrow q7 & 1.0 : patient(q_3, q_5, q_7) \longrightarrow q_8 \\
r(q_8) = 1.0 &
\end{array}$$

This automaton recognizes the first tree of the base on patients of Figure 1:
$t = patient(symptoms(ID1, ID2), blood(A+), stage(severe)))$ with the probability
$p(t \mid A) = 0.48$.

$$\begin{aligned}
p(t \mid A) &= r(\delta'(t)) \times \pi(t) \\
&= r(q_8) \times p(patients, \delta'(symptoms(ID1, ID2)), \delta'(blood(A+)), \delta'(stage(severe))) \times \\
&\quad \pi(symptoms(ID1, ID2)) \times \pi(blood(A+)) \times \pi(stage(severe)) \\
&= 1.0 \times p(patients, q_3, q_5, q_7) \times p(symptoms, \delta'(ID1), \delta'(ID2)) \times \pi(ID1) \times \pi(ID2) \times \\
&\quad p(blood, \delta'(A+)) \times \pi(A+) \times p(stage, \delta'(severe)) \times \pi(severe) \\
&= 1.0 \times 1.0 \times p(patients, q_3, q_5, q_7) \times p(symptoms, q_1, q_2) \times p(ID1) \times p(ID2) \times \\
&\quad p(blood, q_4) \times p(A+) \times p(stage, q_6) \times p(severe) \\
&= 1.0 \times 1.0 \times 1.0 \times 1.0 \times 1.0 \times 1.0 \times 0.6 \times 1.0 \times 0.8 \\
&= 0.48
\end{aligned}$$

An important feature of a stochastic tree automaton is its consistency since it allows to define a statistical distribution over Σ^T, that is:

$$\sum_{t \in \Sigma^T} p(t \mid A) = 1$$

In our inference procedure, probabilities of the SMDTA are computed from random samples and thus consistency is always preserved [10, 35].

Note that these stochastic tree automata are able to recognize ordered trees which are defined on a ranked alphabet (that is where the arity of the nodes are fixed). As far as we know, there are no attempts for learning tree automata with unordered trees (which seems to be a difficult task), only data mining approaches [31] or machine learning of tree pattern languages [3] have tried to investigate this field. However, we can notice that [26] tries to learn unranked tree automata with ordered trees but in a non probabilistic framework. In this paper, we rather focus on learning stochastic tree automata working with ordered trees defined over a ranked alphabet.

5.2. Induction of Stochastic Tree Automata

In stochastic grammatical inference, Carrasco *et al.* [8] proposed an efficient algorithm to learn stochastic tree automata. Abe *et al.* [1] dealt with learning stochastic tree grammars to predict protein secondary structure. Rico *et al.* [33] presented a generalization of k-gram models for stochastic tree languages.

Our inference procedure is an extension of [8], that takes sorts into account [19]. Algorithm 2 gives an idea of the main steps of the inference. For formally details, the interested reader may refer to [8, 19]. The input of the algorithm is a training set T of trees and the output is a SMDTA which respects the distribution over T.

The algorithm computes the transition function considering all subtrees of the training set. A total order is defined on subtrees comparing their depth. Each subtree is mapped to a state, taking into account the fact that if two subtrees are similar in the training set, then they have the same state. We denote by $[t]$ the state mapped to the subtree t. To compute the similarity of two subtrees ($comp$ function), the algorithm uses a statistical test [21] depending on a parameter $0 \leq \beta \leq 1$, which corresponds to the

Data: T: a training set of trees,
$\Sigma = (\tau, V, arity, \sigma)$: a signature
Result: $A = (\Sigma, Q, \delta, p, r)$: a SMDTA
begin
  $W \leftarrow Sub(T)$
  $Q \leftarrow \emptyset$
  while $W \neq \emptyset$ **do**
    $x \leftarrow g(t_1, \ldots, t_n) = min\ W$
    $W \leftarrow W \backslash \{x\}$
    if $\exists y \in Q \mid \sigma(x) = \sigma(y)$ *and* $comp(x, y, \beta)$ **then**
      $\delta(g, [t_1], \ldots, [t_n]) = y$
    else
      $Q \leftarrow Q \cup \{x\}$
      $\delta(g, [t_1], \ldots, [t_n]) = x$
    end
  end
  $compute\_probabilities(T, \delta, p, r)$
end

Algorithm 2: Inference of a SMDTA.

Type I error of a similarity test of two subtrees. Intuitively β represents a tolerance parameter for the merging of two subtrees into a same state. The $comp$ function is described in Algorithm 3. The notation $C_T(t)$ corresponds to the number of occurrences of a subtree t in the learning set T. The similarity of two subtrees is assessed by using the regular tree patterns of Σ_T^X, the test evaluates if two subtrees are statistically equivalent in a same context. The probabilities are then computed counting subtree occurrences in the training set, with respect to the normalization of SMDTA.

Data: x a subtree of $Sub(T)$
y a subtree of $Sub(T)$
β a real $\in [0; 1]$
begin
  foreach $\forall z \in \Sigma_T^X$ *such that* $z._{\#}x \in T$ *or* $z._{\#}y \in T$ **do**
    if $\left| \frac{C_T(z._{\#}x)}{C_T(x)} - \frac{C_T(z._{\#}y)}{C_T(y)} \right| > \sqrt{\frac{1}{2} * \ln\left(\frac{2}{\beta}\right)} * \left(\frac{1}{\sqrt{C_T(x)}} + \frac{1}{\sqrt{C_T(y)}} \right)$ **then return** false
  return true
end

Algorithm 3: The $comp$ function

The algorithm has the following properties: it is polynomial in the number of different subtrees of the training set, and it converges to the limit under the Gold paradigm [17]. If the set T of terms has been generated with a target stochastic tree automaton A_{target} and if $A(T)$ is the automaton learned from T,

then we have the following property:

$$\lim_{|T|\longrightarrow\infty} \sum_{t\in\Sigma^T} \Big| p(t|A_{target}) - p(t|A(T)) \Big| = 0$$

5.3. Evaluation of Stochastic Tree Automata

The evaluation of non-probabilistic models is often based on the correct classification rate. This is not the case for our approach aiming at learning a probability distribution from a learning sample. Probabilistic models are rather assessed on their ability to correctly predict the probability of the examples of a test sample. When the target model is known, the correctness is evaluated by the distance between the statistical distribution of the learned model and the target model [7]. Nevertheless, when we work on real applications, the target model is often unknown, and the quality of the learned model can be evaluated with a measure of perplexity.

5.3.1. Probabilistic Distance

Lyngsø *et al.* [28] defined distances between two hidden Markov models introducing the co-emission probability, as the probability that two independent models generate the same string. Carrasco *et al.* [9] presents an adaptation of the co-emission to stochastic tree automata. The probability that two probabilistic models, A and A', generate the same tree is defined by

$$C(A, A') = \sum_{t\in\Sigma^T} p_A(t) \times p_{A'}(t)$$

Where $P_A(t)$ is the probability of t given the model A. The co-emission probability allows us to define a distance D_a which can be interpreted as the measure of the angle between the vectors representing automata in a space whose base is the set of trees of Σ^T.

Definition 16. The distance D_a between two automata $A1$ and $A2$ is defined by:

$$D_a(A1, A2) = \arccos\left(\frac{C(A1, A2)}{\sqrt{C(A1, A1) * C(A2, A2)}}\right)$$

In the context of stochastic automata, Carrasco *et al.* [9] proposed a recursive definition of the co-emission in order to compute a distance between two stochastic automata. Then, the co-emission can be written such that

$$C(A, A') = \sum_{q\in Q} \sum_{q'\in Q} r_A(q) \times r_{A'}(q') \times \nu_{q,q'}$$

where

$$\nu_{q,q'} = \sum_{\substack{f\in\Sigma \ s.t. \\ f(q_{i1},\ldots,q_{im})\to q, \\ f(q_{j1},\ldots,q_{jm})\to q'}} p(f(q_{i1},\ldots,q_{im}) \to q) \times p'(f(q_{i1},\ldots,q_{im}) \to q) \times \nu_{qi1,qj1} \times \ldots \times \nu_{qim,qjm}$$

The recursive definition of $\nu_{q,q'}$ stops for constant symbols arriving on q or q'.

5.3.2. Perplexity

In the case of stochastic tree automata the quality of a model A on a test set S_{test} can be evaluated by the following measure:

$$LL = \frac{1}{\| S_{test} \|} \sum_{t \in S_{test}} \log p(t \mid A)$$

where $\| S_{test} \|$ is the number of nodes of every tree of S_{test}.

A perfect model can predict each element of the test set with a probability equal to one, and so $LL = 0$. In a general way, we consider the perplexity of the test set which is defined by $PP = 2^{LL}$. A minimal perplexity ($PP = 1$) is reached when the model can predict each element of the test sample. Therefore we consider that a model is more predictive than another if its perplexity is lower.

A problem occurs when a tree of the test sample cannot be recognized by the automaton. Actually the probability of this example is 0 and the perplexity cannot be computed. To avoid this problem, a classical method consists in smoothing the distribution of the learned model using an interpolation approach [30] with a unigram model A_0 recognizing all trees of Σ^T. This automaton has only one state, and for each function symbol of the signature, there is a transition starting from and ending in this state. Finally, a transition is added for any unknown symbol. The probabilities of transition rules are computed from the training set, keeping a small ratio for the transition involving unknown symbols. In the smoothed model, a tree t has the probability:

$$\hat{P}(t) = \lambda.p(t|A) + (1 - \lambda).p(t|A_0)$$

where $0 \leq \lambda \leq 1$.

5.4. Experimentations

In this section, we present a set of experiments showing the interest of our method in the context of learning in presence of noisy data. We study the ability of our method to remove irrelevant or noisy subtrees to learn stochastic tree automata from a set of trees.

To verify this behavior, we carried out two series of experiments. First, we consider datasets where the target automaton is known and we evaluate the quality of the learned model by computing the D_a distance. For this experiment, we used five artificial datasets.

- The dataset Stacks is produced by an automaton representing stacks of objects (squares and triangles) [4]. Each object is described by its shape and its color. This dataset has an alphabet constituted of 8 different leaves and allows to generate trees with a comb pattern representing the size of the stack. A sample of 5000 examples is generated for this dataset.

- The dataset Bool is generated by an automaton modeling boolean expressions. The alphabet includes only two leaves (*true* and *false*) and the generated trees define boolean expressions with operators *and*, *or* and *not*. We use a sample of 5000 trees for this dataset.

- The dataset Cond is defined an automaton representing conditional statements of a programming language [8]. This dataset has an alphabet with 13 leaves and allows to generate wide and depth trees. We infer automata from a sample of 3000 trees.

- The datasets Art1 and Art2, are generated by automata represented in Figure 6, and define simple languages with different leaves. In these datasets, the location of the leaves has an important effect on the language. In this case, each dataset is composed of 5000 trees.

0.40: a→q1	0.50: f(q1,q1,q2)→q3
0.30: b→q1	0.50: g(q2,q2,q1)→q4
0.30: c→q1	0.30: f(q1,q4,q2)→q3
0.55: d→q2	0.30: g(q2,q3,q1)→q4
0.45: e→q2	0.20: f(q1,q3,q2)→q3
0.20: g(q2,q4,q1)→q4	Final State r(q4)=1

(a) Automaton of Art1.

0.40: a→q1	0.60: f(q1,q1,q2)→q3
0.30: b→q1	0.60: g(q2,q2,q1)→q4
0.30: c→q1	0.40: f(q1,q4,q2)→q3
0.55: d→q2	0.40: g(q2,q3,q1)→q4
0.45: e→q2	Final State r(q4)=1

(b) Automaton of Art2.

Figure 6. Databases Art1 and Art2.

The other experiments concern samples for which we do not know the target model. Then we measure the quality of the learned models by the perplexity measure. In this context, we re-use the five previous samples and we also evaluate our approach on 3 real datasets.

- The database Loan of the UCI Irvine [5]. This dataset corresponds to structured data in 1st order logic and is converted into trees according to the principle presented in [4]. This sample has 1000 examples composed of 143 different leaves.

- The database Tacrine comes from a dataset on the toxicity of the tacrine molecule presented in [24]. This dataset is also structured in 1st order logic and is also converted into trees with the principle proposed in [4]. This dataset has 1000 examples with 42 different leaves, but has larger trees than the previous one.

- We also use a sample proposed for the PKDD'02 discovery challenge [1] (dataset on hepatitis); the transformation of data into trees is described in [19]. This dataset has 4000 examples with 253 leaves.

Our experimental setup consists in adding noise in each dataset, as previously described in Section 4, with a level γ from 0 to 25%. For experiments with an unknown target, we use a 5 fold cross-validation, where only the learning sample is noisy. Then we preprocess each corrupted sample with our approach using a Type I error α between 0.25 and 0.01. For each level of noise, we keep the α value which minimizes the quality measure. In order to show the efficiency of our pruning technique, we compare it with a Monte-Carlo sampling which randomly removes $\alpha\%$ of the subtrees.

In our experiments, we study two variants of our pruning method. In the first one (called M_1), the removed subtree is replaced by a special symbol that does not appear in the dataset. This way to proceed looks like to the one used in classic data reduction techniques. The other one (called M_2) consists in

[1] http://lisp.vse.cz/challenge/ecmlpkdd2002/

replacing a deleted subtree by the most probable one of the partition, in order not to lose a part of the information. In the Monte-Carlo approach, this subtree is chosen randomly among the other subtrees.

We use two statistical measures to evaluate the results: the mean and the standard deviation of the quality criterion (distance or perplexity) for the different levels of noise.

The results are presented in Tables 1, 2 and 3. In each of them, we indicate the pruning rate according to the number of nodes removed in column Red. The global size (in the total number of nodes) of the datasets is indicated by IS. The results obtained without a preprocessing are stored in a column with a suffix N (the prefix corresponds to the mean for the criteria D_a and $Perp$), those obtained by our approach with CI. The column MC corresponds to the Monte-Carlo sampling method previously mentioned. Finally, the column Sig indicates the significance of the results between N and CI using a Student paired t-test over the means with a critical risk of 5%. The column $Sig2$ indicates the significance of the results between CI and MC using the same comparison test.

5.4.1. Experimentations Knowing the Target Automaton

Before detailing the behavior of our pruning method, we present in Figure 7, the evolution of the mean of variances, before the preprocessing, for each dataset. The interest of this experiment is to empirically show the effect of the Theorem 4.2. Actually, the database Bool, which presents a steady variance, is the one having the smallest alphabet size. It confirms that the probabilistic distribution has not been changed a lot in the presence of noise, that has a direct impact on the pruning method. On the other hand, the databases Stacks and Cond, which have a higher alphabet size, present a more disturbed variance.

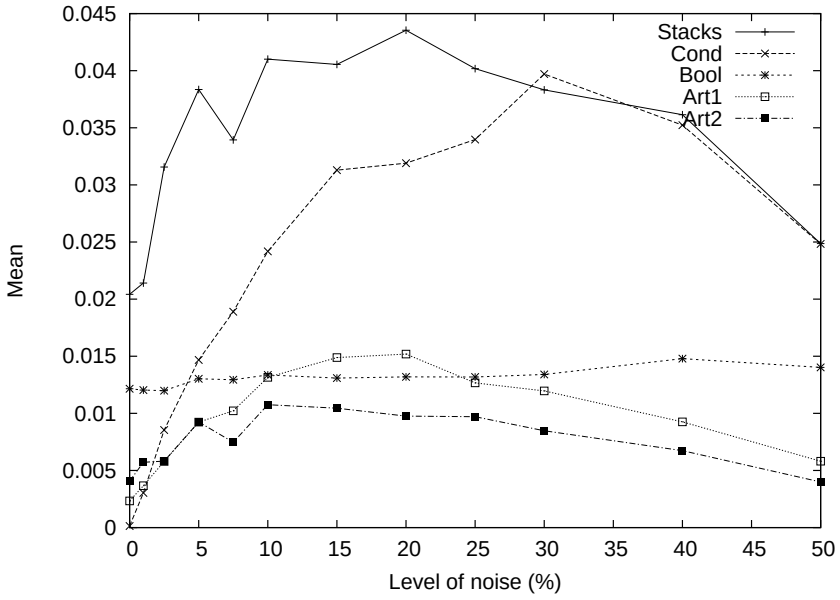


Figure 7. Mean of variances of probabilities for the 5 artificial datasets.

Table 1 synthesizes results for the variant M_1 which replaces, as we said before, the deleted subtree by a special symbol. The results for the variant M_2, which replaces the deleted subtree by the most

probable one, are presented in Table 2. We give the average distance $\overline{D_a}$ observed between the target automaton and the inferred automaton $\pm$ the standard deviation.

$Base$	IS	Red	$\overline{D_a.N}$	$\overline{D_a.CI}$	$\overline{D_a.MC}$	Sig	$Sig2$
Stacks	35724	3.5%	0.30±0.11	0.23±0.15	0.60±0.45	yes	yes
Cond	71683	14%	0.60±0.22	0.44±0.12	1.00±0.18	yes	yes
Bool	43185	76%	0.22±0.08	0.16±0.09	0.17±0.10	yes	yes
Art1	33137	8.6%	1.26 ±0.28	0.51±0.20	0.56±0.21	yes	yes
Art2	30113	1.3%	0.19±0.15	0.16±0.15	0.18±0.15	yes	yes
Average	42678	27%	0.51 ±0.16	0.30±0.14	0.35±0.22	yes	yes

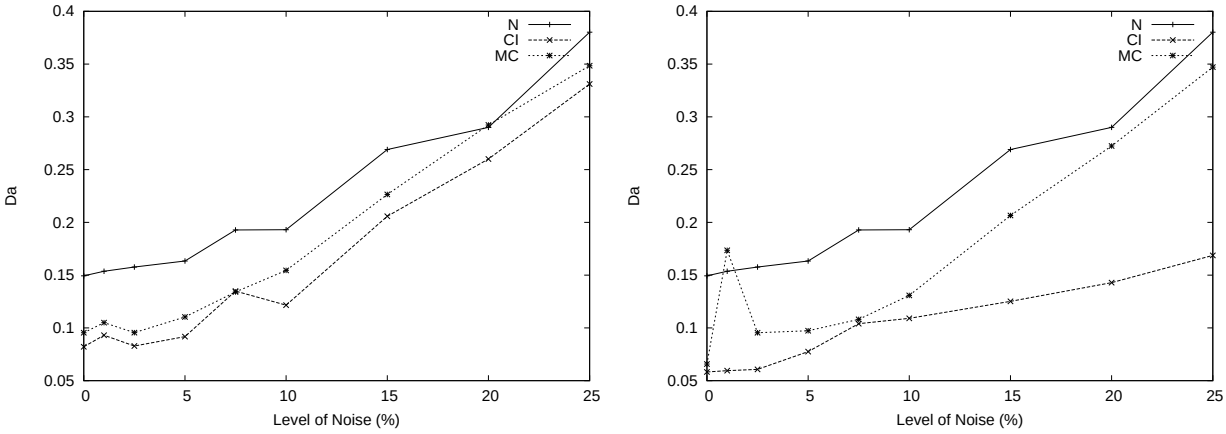
Table 1. Distances D_a to the target automaton with the variant M_1.

$Base$	IS	Red	$\overline{D_a.N}$	$\overline{D_a.CI}$	$\overline{D_a.MC}$	Sig	$Sig2$
Stacks	35724	1.4%	0.30±0.11	0.11±0.10	0.13±0.08	yes	yes
Cond	71683	6%	0.60±0.22	0.59±0.23	0.57±0.07	yes	yes
Bool	43185	61%	0.22±0.08	0.10±0.04	0.17±0.09	yes	yes
Art1	33137	1.7%	1.26 ±0.28	0.38±0.06	0.39±0.07	yes	no
Art2	30113	1.1%	0.19±0.15	0.17±0.15	0.18±0.15	yes	yes
Average	42678	14%	0.51±0.11	0.27±0.12	0.29±0.09	yes	yes

Table 2. Distances D_a to the target automaton with the variant M_2.

The first point to note is the fact that our approach is useful for inferring better models in presence of noise, than an approach without preprocessing. Actually, this behavior is true for M_1 and M_2, but the latter seems to have better results. We can easily explain this behavior by the fact that our learning algorithm (here a stochastic automata learning algorithm) is based on the statistical information contained in the dataset. Then, it requires a lot of examples to refine the estimations. This remark is very important because it expresses a particularity of the data reduction techniques that can be used for improving stochastic automata learning algorithms. In the context of probability distribution estimation, our experiments show that a radical deletion is not the optimal way to proceed, and that the replacement by a relevant tree (the most probable one) is more performing. This phenomenon is not surprising, and this kind of approach has already shown its efficiency in Data Mining for dealing with missing attributes. Actually, in the specific field of grammatical inference, a minimal number of examples is needed in order to infer the correct structure of the automaton [14]. Consequently, the suppression of the structure of some examples can affect the quality of the learned automaton and then the quality of the probability distribution inferred.

We can finally notice that the significance tests between N and CI are all in favor of our approach that confirms its interest for preprocessing noisy data. Moreover, those between CI and MC are in favor



(a) Variant M_1.

(b) Variant M_2.

Figure 8. Behavior on the Bool database.

of our approach, except for the base Art1 with the method M_2.

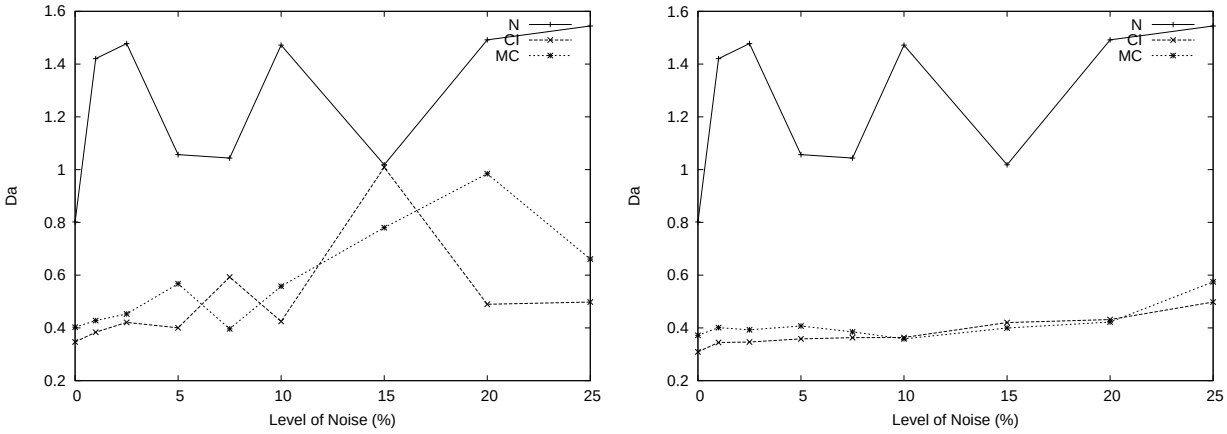
Since Tables 1 and 2 are synthetic, we also present the behavior of our method for the specific base Bool in Figure 8. We can make the following remarks. For both of the alternative variants M_1 and M_2, CI allows to infer automata that are always significantly better than the ones learned from noisy data without preprocessing. Moreover, CI is always better than the Monte-Carlo approach that justifies the claim that we are able to detect and remove noisy subtrees.

The base Bool is an example of a correct behavior of our approach. However this is not always so significant. For example, let consider the behavior of our approach on the base Art1, represented in Figure 9. Our two alternative procedures give good results until a level of noise of 7.5%. Beyond this point, MC has a behavior similar and sometimes better, which shows that our approach has some difficulties to deal with high levels of noise.

An explanation of this phenomenon is that the presence of noise tends to smooth the repartition of the data. Actually, we noticed that when the level of noise increases, the average number of subtrees in a partition tends to decrease. In fact, there are more partitions and less examples in each partition. For example, in the case of the Art1 database, there is an average of 5.37 subtrees in a partition with 1% of noise and only 2.36 subtrees when the noise level is at 10%. In this context, when the noise level is increasing, the estimations of the bounds of the confidence interval are sharpness, resulting in larger intervals. Thus, it is more difficult to detect and remove irrelevant instances. On the other hand, MC always removes a given proportion of trees, whatever the level of noise is. Then, when this level is high, the probability that MC removes a corrupted subtree increases and allows this approach to have a better behavior.

5.4.2. Experimentations without Knowing the Target Automaton

In this context, we divide the sample of trees in two sets: a training set and a test set. Since we want to evaluate our approach in the context of noise, we only add noise to the training set. From this one, we



(a) Variant M_1.

(b) Variant M_2.

Figure 9. Behavior on the Art1 database.

infer two automata: the first one is learned without preprocessing, the second one is inferred after the pruning process. The noise-free test set is then parsed by the two automata resulting in the computation of two perplexity measures. Note that all the experiments are achieved using a 5 fold cross-validation.

$Base$	IS	Red	$\overline{Perp.N}$	$\overline{Perp.CI}$	$\overline{Perp.MC}$	Sig	$Sig2$
Stacks	35724	2%	1.93±0.15	1.83±0.07	1.86±0.08	yes	yes
Cond	71683	3.9%	1.47±0.21	1.33±0.07	1.42±0.10	yes	yes
Bool	43185	3.8%	3.41±0.16	3.56±0.09	3.95±0.21	yes	yes
Art1	33137	2.5%	3.68±0.43	2.50±0.12	2.57±0.15	yes	yes
Art2	30113	1.8%	1.33±0.12	1.30±0.10	1.32±0.11	yes	no
Loan	14117	1%	9.02±2.63	5.10±1.30	5.75±0.49	yes	yes
Tacrine	41130	2.6%	4.60±2.84	3.14±0.77	3.21±0.80	yes	yes
PKDD'02	115173	20%	12.2±4.60	8.75±2.50	9.00±2.00	yes	no
Average	47999	4.7%	4.71±1.39	3.44±0.63	3.64±0.49	yes	yes

Table 3. Perplexity results obtained with the variant M_2.

Since the variant M_2 consisting in replacing a noisy subtree by the most probable has already shown its efficiency with the D_a criterion, we only test this approach for the second series of experiments. The results are presented in Table 3. From a general point of view, these results confirm the behavior already seen in the previous experiments. Our pruning method CI permits to efficiently preprocess all the databases except the base Bool. Moreover, an interesting remark is that our approach has a smaller standard deviation than the one computed from the unpreprocessed sets, that denotes a better robustness.

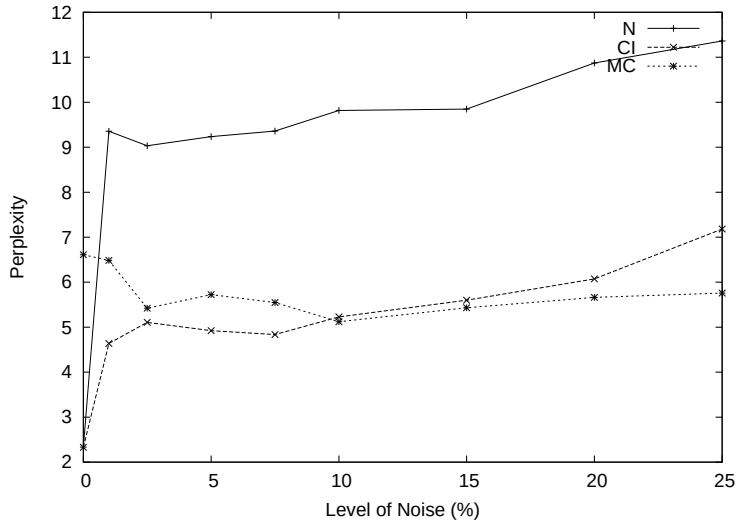


Figure 10. Behavior for Loan with perplexity and for the variant M_2.

One of the interest of this series of experiments is also to observe the behavior of CI on real datasets. The results are always in favor of our method, but the most interesting effect is certainly obtained for the base PKDD'02, with a high reduction rate and a large decrease of the perplexity. As for the previous criterion D_a, let us study the behavior of CI on a single database (here the base Loan, see Figure 10). Whatever the level of noise, CI dramatically decreases the perplexity in comparison with the unpre-processed dataset. Moreover, the level of the deviation between the curves N and CI is kept with the increase of the noise. In comparison with the Monte-Carlo sampling, CI provides better results until 10% of noise. Beyond this rate, it begins to have some difficulties to remain performing, which confirms the fact our approach is not very suited for dealing with high levels of noise.

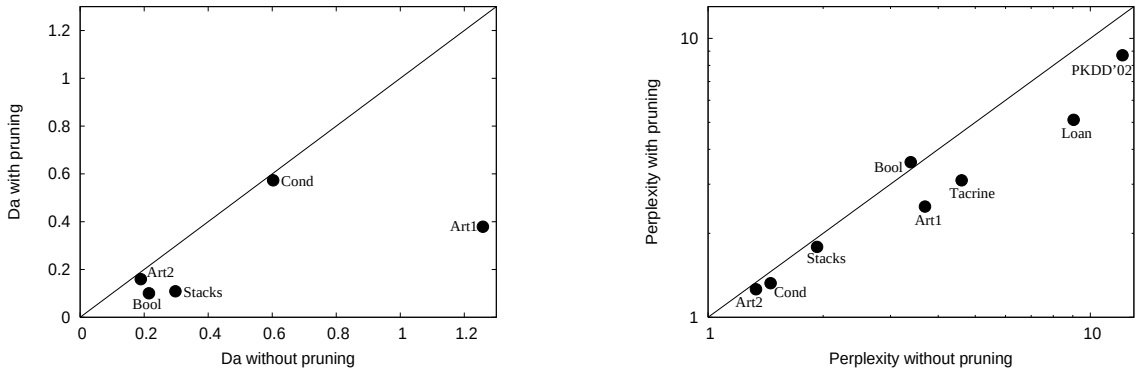


Figure 11. Results of the experimentations obtained with the variant M_2.

In order to compare the two measures we used for evaluating the performances, we summarize all

our results, obtained with the variant M_2, in Figure 11. Each dot represents a database. A dot under the bisecting line expresses the fact that our pruning approach CI is better than the one without pruning.

6. Conclusion

In this paper we have presented an original approach allowing to deal with irrelevant and noisy subtrees in a set of trees. This approach is a data reduction technique that can be considered as a hybrid approach. It can actually delete either a whole irrelevant tree or only some of its subtrees. Our pruning method is based on the use of confidence intervals computed from a probability distribution on subtrees appearing in a same context. Our experiments have shown that our approach is efficient and robust resulting in the inference of automata closer to the target concept in the presence of noisy data.

Despite this interesting general behavior, we have experimentally noted that our pruning method has some difficulties to deal with datasets with a small alphabet like the one of boolean expressions or parity functions. Moreover, our approach is based on confidence intervals which depend on the size of the learning set. Then, its efficiency is only ensured with large databases, that is not a too strong hypothesis considering the size of the modern databases. Finally, our method depends on the partitioning technique for which we propose a first solution based on the notion of context. We think that this solution requires a relatively strong constraint on the trees. Indeed, when there is a too small number of elements in a partition, the computation of the confidence intervals is inaccurate and our approach becomes less efficient, especially for high levels of noise. We are working on other relaxing partitioning approaches which deserve further investigations. Finally, we think that an interesting improvement of our method would concern its adaptation to unordered trees, which are for example suited for dealing with XML data.

Acknowledgment

This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

References

[1] Abe, N., Mamitsuka, H.: Predicting protein secondary structure using stochastic tree grammars, *Machine Learning*, **29**, 1997, 275–301.

[2] Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB* (J. B. Bocca, M. Jarke, C. Zaniolo, Eds.), Morgan Kaufmann, 12–15 1994, ISBN 1-55860-153-8.

[3] Amoth, T. R., Cull, P., Tadepalli, P.: On Exact Learning of Unordered Tree Patterns, *Machine Learning*, **44**(3), 2001, 211–243.

[4] Bernard, M., de la Higuera, C.: GIFT: Grammatical Inference for Terms, *Late Breaking paper at the 9th International Conference on Inductive Logic Programming*, June 1999.

[5] Blake, C., Merz, C.: University of California Irvine Repository of machine learning databases, `http://www.ics.uci.edu/~mlearn/`, 1998.

[6] Brown, P., Pietra, V. D., deSouza, P., Lai, J., Mercer, R.: Class-Based n-gram Models of Natural Language, *Computational Linguistics*, **18**(4), 1992, 467–479.

[7] Calera-Rubio, J., Carrasco, R. C.: Computing the relative entropy between regular tree languages, *Information Processing Letters*, **68**(6), 1998, 283–289.

[8] Carrasco, R., Oncina, J., Calera-Rubio, J.: Stochastic Inference of Regular Tree Languages, *Machine Learning*, **44**(1/2), 2001, 185–197.

[9] Carrasco, R. C., Rico-Juan, J. R.: A similarity between probabilistic tree languages: application to XML document families, *Pattern Recognition, in press*, 2002.

[10] Chaudhuri, R., Rao, A. N. V.: Approximating Grammar Probabilities: Solution of a Conjecture, *Journal of the Association for Computing Machinery*, **33**(4), 1986, 702–705.

[11] Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree Automata Techniques and Applications , Available on: `http://www.grappa.univ-lille3.fr/tata`, 1997.

[12] Cover, T. M., Hart, P. E.: Nearest Neighbor Pattern Classification, *IEEE Transactions on Information Theory*, **13**(1), 1967, 21–27.

[13] De Raedt, L.: Data mining in Multi-Relational Databases, *4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2000)*, 2000, Invited talk.

[14] Dupont, P., Miclet, L., Vidal, E.: What Is the Search Space of the Regular Inference?, *Proceedings of the Second International Colloquium on Grammatical Inference and Applications*, number 862 in LNAI, Springer-Verlag, 1994.

[15] Garcia, P., Oncina, J.: *Inference of recognizable tree sets*, Research Report DSIC - II/47/93, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, 1993.

[16] Gécseg, F., Steinby, M.: *Tree Automata*, Akadémiai Kiadó, Budapest, 1984.

[17] Gold, E. M.: Language Identification in the limit, *Information and Control*, **10**(n5), 1967, 447–474.

[18] Goldman, S. A., Kwek, S. S.: On Learning Unions of Pattern Languages and Tree Patterns, *10th Algorithmic Learning Theory conference*, 1720, 1999.

[19] Habrard, A., Bernard, M., Jacquenet, F.: Generalized stochastic tree automata for multi-relational data mining, *6th International Colloquium on Grammatical Inference. ICGI 2002*, 2484, Springer, Amsterdam, september 2002.

[20] Habrard, A., Bernard, M., Jacquenet, F.: Multi-Relational Data Mining in Medical Databases, *Proceedings of the 9th Conference on Artificial Intelligence for Medicine Europe (AIME'03)* (M. Dojat, E. Keravnou, P. Barahona, Eds.), 2780, Springer–Verlag, Protaras, Cyprus, October 2003.

[21] Hoeffding, W.: Probability inequalities for sums of bounded random variables, *Journal of the American Statistical Association*, **58**(301), 1963, 13–30.

[22] John, G., Kohavi, R., Pfleger, K.: Irrelevant Features and the Subset Selection Problem, *11th International Conference on Machine Learning*, 1994.

[23] Kilpeläinen, P.: *Tree Matching Problems with Applications to Structured Text Databases*, Report a-1992-6, Department of Computer Science, University of Helsinki, November 1992.

[24] King, R., Srinivasan, A., Sternberg, M.: Relating chemical activity to structure: An examination of ILP successes, *New Generation Computing, Special issue on Inductive Logic Programming*, **13**(3-4), 1995, 411–434.

[25] Knuutila, T., Steinby, M.: Inference of tree languages from a finite sample: an algebraic approach, *Theoretical Computer Science*, **129**, 1994, 337–367.

[26] Kosala, R., Bruynooghe, M., den Bussche, J. V., Blockeel, H.: Information Extraction from web documents based on local unranked tree automaton inference, *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-2003)*, 2003.

[27] Kosala, R., Bussche, J., Bruynooghe, M., Blockeel, H.: Information Extraction in Structured Documents using Tree Automata Induction, *6th European Conference on Principles and Practise of Knowledge Discovery in Databases (PKDD'02)*, 2431, Springer, 2002.

[28] Lyngsø, R., Pedersen, C., Nielsen, H.: Metrics and Similarity Measures for Hidden Markov Models, *7th International Conference on Intelligent Systems for Molecular Biology, ISMB '99 Proceedings*, AAAI Press USA, Heidelberg, Germany, 1999.

[29] Miyahara, T., Suzuki, Y., Shoudai, T., Uchida, T., Takahashi, K., Ueda, H.: Discovery of Frequent Tag Tree Patterns in Semistructured Web Documents, *PAKDD 2002*, Taipei, Taiwan, May 2002.

[30] Ney, H., Essen, U., Kneser, R.: On Structuring Probabilistic Dependences in Stochastic Language Modelling, *Computer Speech and Language*, **8**, 1994, 1–38.

[31] Nijssen, S., Kok, J. N.: Efficient Discovery of Frequent Unordered Trees, *Proceedings of the First International Workshop on Mining Graphs, Trees and Sequences (MGTS-2003)* (L. de Raedt, T. Washio, Eds.), september 2003, ECML/PKDD'03 workshop proceedings.

[32] Nock, R., Sebban, M.: Sharper Bounds for the Hardness of Prototype and Feature Selection, *International Conference on Algorithmic Learning Theory* (S. V. L. 1968, Ed.), 2000.

[33] Rico-Juan, J., Calera, J., Carrasco, R.: Probabilistic k-Testable Tree-Languages, *ICGI 2000, Lisbon (Portugal)* (A. L. Oliveira, Ed.), 1891, Springer, Berlin, September 2000.

[34] Rico-Juan, J., Calera-Rubio, J., Carrasco, R.: Stochastic k-testable Tree Languages and Applications, *ICGI 2002*, 2484, Springer-Verlag, Amsterdam (Nederland), september 2002.

[35] Sakakibara, Y.: Efficient Learning of Context-Free Grammars from Positive Structural Examples, *Information and Computation*, **97**, 1992, 23–60.

[36] Termier, A., Rousset, M., Sebag, M.: TreeFinder: a First Step towards XML Data Mining, *International Conference on Data Mining (ICDM'02)*, 2002.

[37] Wilson, D., Martinez, T.: Reduction Techniques for Instance-Based Learning Algorithms, *Machine Learning*, **38**(3), 2000, 257–286.

[38] Zaki, M.: Efficiently mining frequent trees in a forest, *Proceedings of the 8th International Conference on Knowledge Discovery and Data Mining (KDD)*, 2002.